report

I have used merge sort for this problem as merge is faster and merge sort also uses similar logic as in this problem like splitting the array into two and sorting them and merging two sorted arrays

The peak part of this programme is at wrriting the void * function of merge sort for sending into pthread_create

This was made possible by ***casting*** here i used a trick and casted the array pointer into void pointer and passed it into the function as argument and later in the function i recasted it into array pointer by this way I was able to achieve it

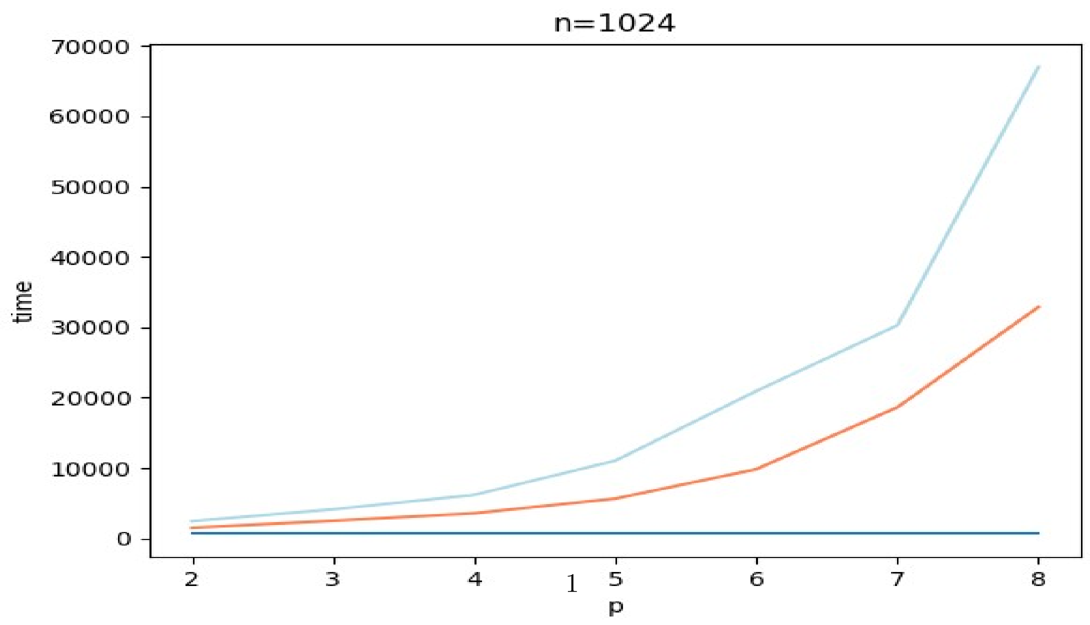
The **void** merge sort function will say the intial and final points of array that the thread should sort

In first method after the sorting of threads the main thread sorts everything in a proper manner here we used a function merge which is already present in merge_sort

In method 2 the sorted array are merge using slave threads here I called everything as a slave threads and assigned the generation of slaves so that each generation knows how much chunk of the array It need to be sorted

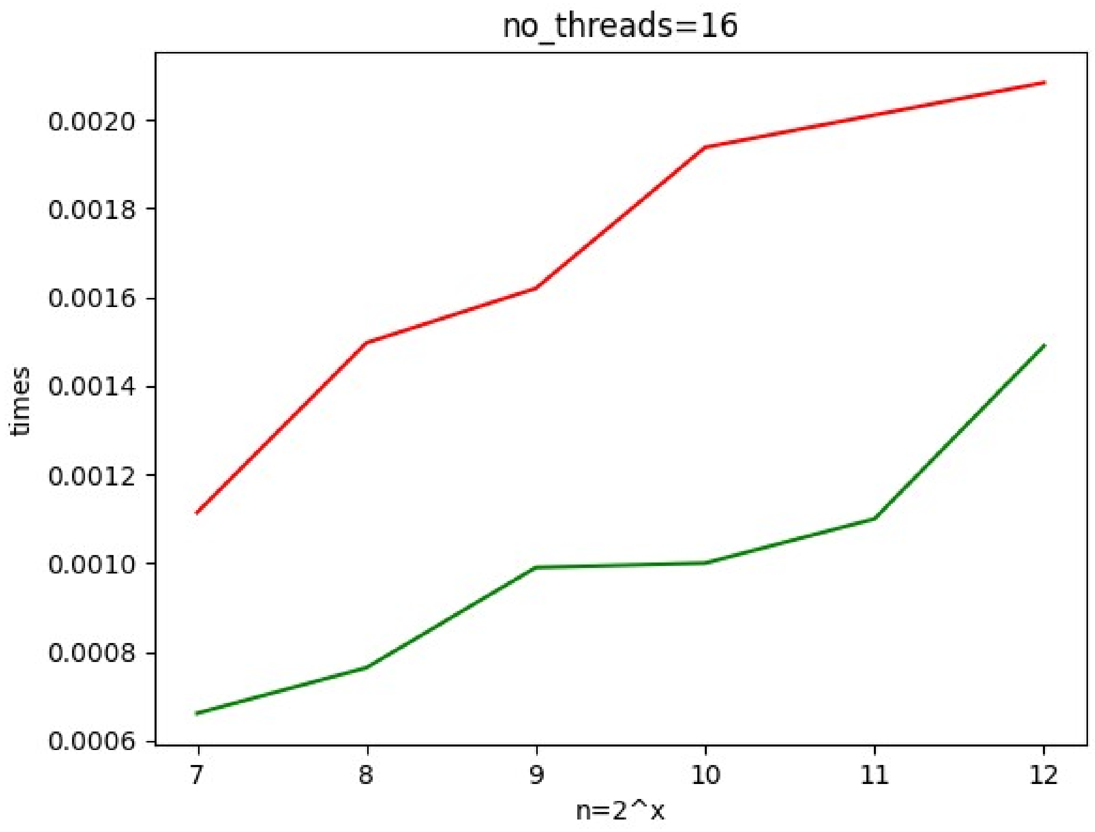
The main Issue with second method we are creating and destroyind threads so even though it is parllel the time taken by it is much more than sequential rather I f we reuse the already created threads the we can notice a siginificant time difference between sequential and parllel threads

Graphs:



In this graph the light blue one is mth2 and red one is mth1 and and solid blue is sequential mergesort

This is the one of the few best counter examples where threading increase the time as the merge sorting is very fast it’s tiime is very comparabe for creation of the threads so this is taking lot of time as we increase number of threads



This is the common graph as we know that as increase size of the array it takes time to sort the array here red one is mth2 and green one is mth1

<u>Conclusion :</u>

By this we can understand that thre is no use for having multiple threads for very rapid process